Attention-Based Object Detection

Manar Abdelatty, Zainab Iftikhar, Titas Grusnius, Tyler Jacobson
CSCI 1470 – Deep Learning Day, Spring 2022

Introduction

Object detection algorithms are critically important for applications as diverse as self-driving cars, face recognition, motion tracking, and pedestrian monitoring, and therefore it is critical to develop object detection algorithms that are as accurate yet architecturally simple and scalable as possible. This project attempts to reimplement the novel object tection algorithm presented by Carion et al. in the paper "End-to-End Object Detection with Transformers" [1]. The model produces a set of bounding boxes and category labels around objects detected in images using a CNN-encoder-decoder backbone.

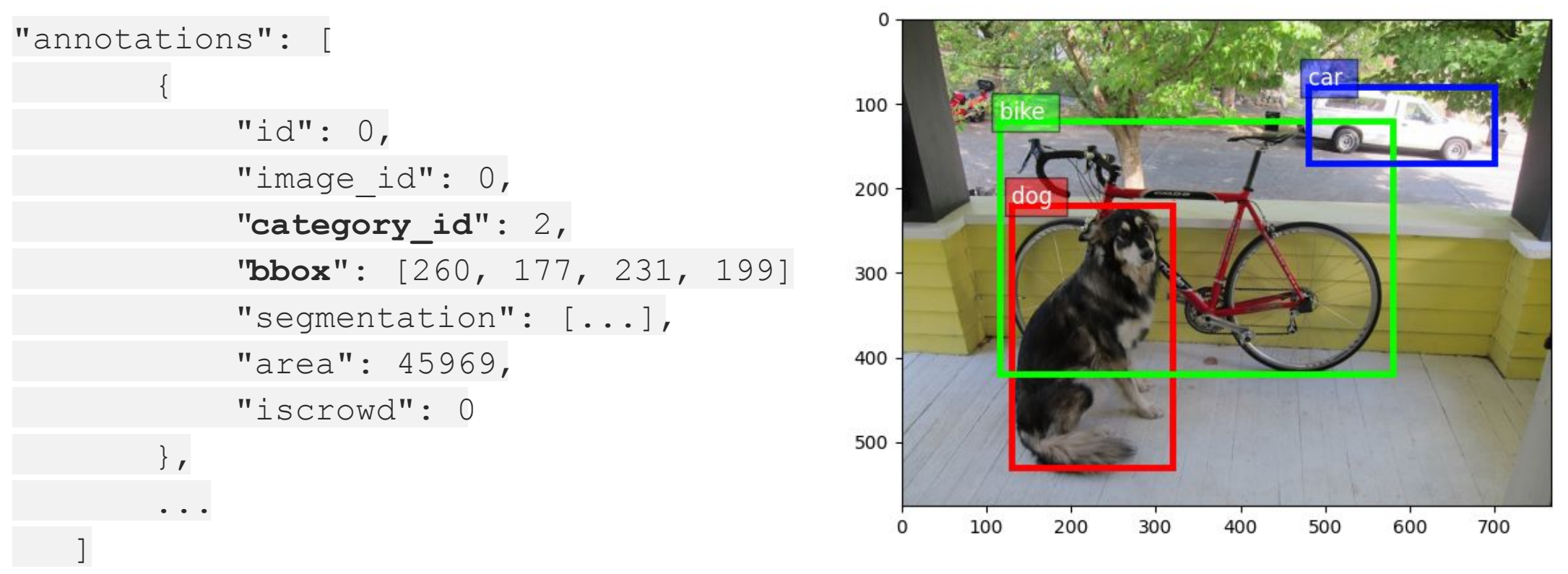
Dataset

This model uses the Microsoft Common Objects in Context (COCO) 2017 detection dataset.

- 118K Training Images
- 5K Validation Images

The ground truth labels for the object detection task are:

- Bounding box coordinates (x, y, w, h)
- Class Labels for each bounding box.



```
"annotations": [
        {
            "id": 0,
            "image_id": 0,
            "category_id": 2,
            "bbox": [260, 177, 231, 199]
            "segmentation": [...],
            "area": 45969,
            "iscrowd": 0
        },
        ...
    ]
```

Figure 1: Sample data point from the COCO 2017 dataset

- Each image has at least one bounding box annotation with an average of 7 bounding boxes per image and a maximum of 63.
- There are a total of 92 object categories in the dataset.

Methodology

- The model relies on a framework called a DEtection TRansformer (DETR). It is composed of:
  1) CNN backbone to produce a set of image features.
  2) Transformer encoder and decoder.
  3) 3-layer prediction feed-forward network with a ReLU activation and a linear projection layer with a softmax activation.
- The model outputs a list of N tuples, where each tuple contains predicted bounding box coordinates and the corresponding class label.

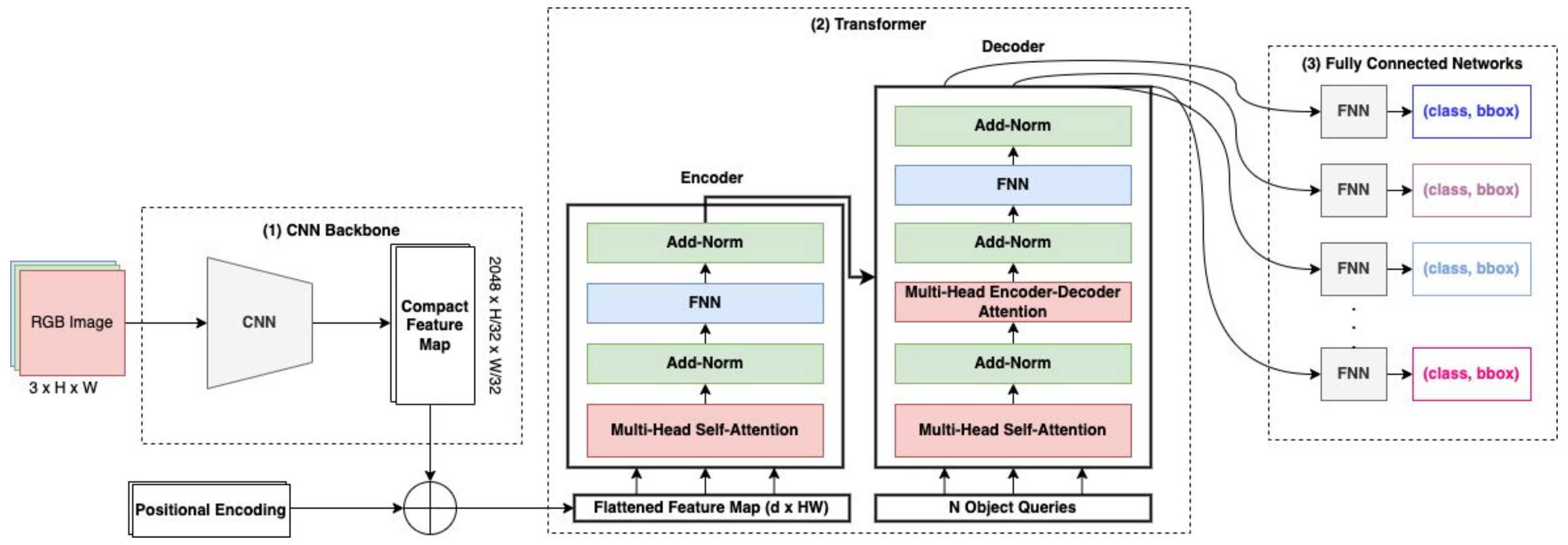


Figure 2: DeTR Model Architecture: A CNN backbone followed by a transformer and a fully connected network [1]

- The model uses a bipartite matching function and Hungarian loss to train the model.
  - ○ The bipartite matching function assigns prediction to ground truth boxes.

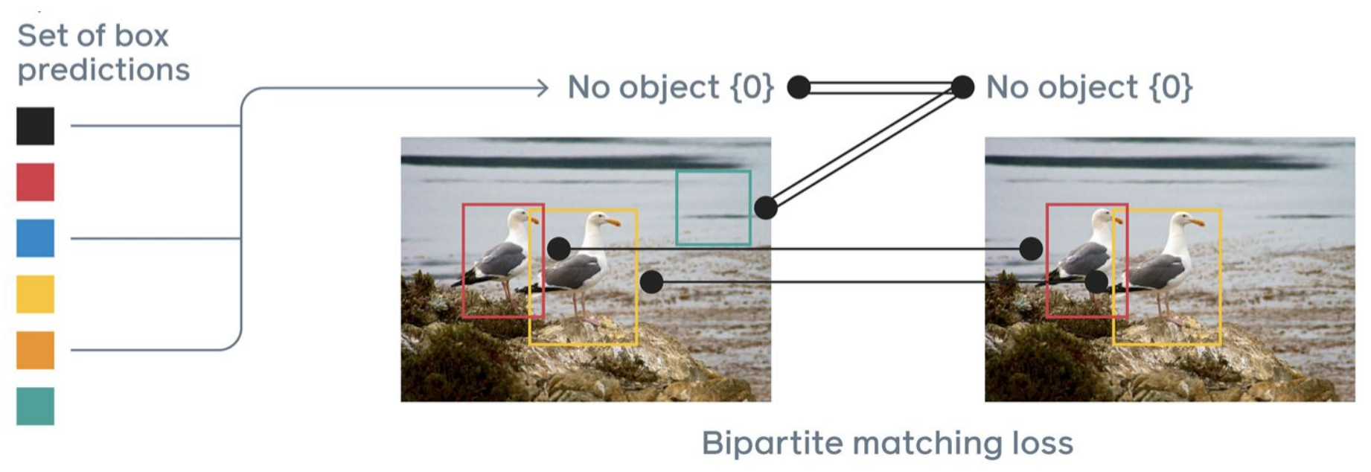
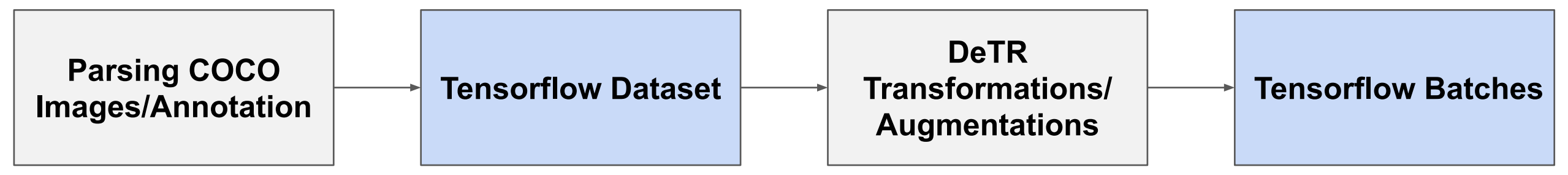


Figure 3:bipartite matching function [1]

  - ○ The hungarian loss is defined as a weighted linear combination of a negative log-likelihood for class prediction and a box loss.

Results

- We were able to get the following parts of the codebase up and running in tensorflow:
  - ■ Data Pipeline



Parsing COCO Images/Annotation
Tensorflow Dataset
DeTR Transformations/ Augmentations
Tensorflow Batches

  - ■ Forward Pass
    - Includes running the input batches through the CNN backbone, transformer block, and fully connected layer.
  - ■ Loss Computation
    - Includes bipartite matching function & Hungarian Loss.
- The Hungarian matcher uses non-differentiable tensorflow operation, namely Scipy *linear_sum_assignment* function which caused gradients to be always *None*.

Discussion

- The model that we tried to re-implement in this project is suitable only for inference or training with distributed/parallel training. The DeTR framework is relatively recent with multiple aspects of novelty which made the pytorch-tensorflow translation quite challenging.

References

[1] Carion, Nicolas, et al. "End-to-end object detection with transformers." *European conference on computer vision*. Springer, Cham, 2020.

[2] Caesar, Holger, Jasper Uijlings, and Vittorio Ferrari. "Coco-stuff: Thing and stuff classes in context." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018.